ENKBF·2.0

VAX 11/730 WCS
MARCH TEST

EP-ENKBF-DL-2.0
FICHE 1 OF 1

JAN 1983
COPYRIGHT© 82-83
MADE IN USA

digital

IDENTIFICATION

Product code: ENKBF VERSION 2.0

Product name: OPTIONAL 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS AND CPU MODULES

Product date: 11-OCT-82

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 3
5.1 Options 3
5.2 Event Flags 4
5.3 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
6.1 Program Overview 4
6.2 Program Size 4
6.3 Program Run Times 4
6.4 Run-time Dynamics 4
6.5 Fault Detection 5
6.6 Performance During Hardware Failures 5
6.7 Program Applications 5
6.8 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is an OPTIONAL 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This OPTIONAL module can be run anytime after module ENKBA is run. This module contains a WCS march pattern test which gives better coverage than the two WCS tests contained in module ENKBA.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBF after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following descibes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 1K byte long.

6.3 Program Run Times

This program will take approximately 2 minutes and 55 seconds to run on a system with 17K of WCS memory.

6.4 Run-time Dynamics

The program will take approximately 20 seconds longer to run for each additional 1K of wcs memory.

6.5 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKBF)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatable.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
11-OCT-82	2.0	RD changes.

```
7000     3
7000     4
7000     5
7000     6
7000     7
7000     8
7000     9
7000    10
7000    11
7000    12
7000    13
7000    14
7000    15
7000    16
7000    17
7000    18
7000    19
7000    20
7000    21
7000    22
7000    23
7000    24
7000    25
7000    26
7000    27
7000    28
7000    29
7000    30
7000    31
7000    32
7000    33
7000    34
7000    35
7000    36
7000    37
7000    38
7000    39
7000    40
7000    41
7000    42
7000    43
7000    44
7000    45
7000    46
7000    47
7000    48
7000    49
7000    50
7000    51
7000    52
7000    53
7000    54
7000    55
7000    56
7000    57
7000    58
7000    59
7000    60
7000    61
7000    62
```

```
;       ENKBF  8085 based diagnostic for WCS/DAP module Rev 02.00
;       Table of Contents
;
;  133  DATA AREA
;  301  PROGRAM ENTRY POINTER
;  320  VARIABLES AREA
;  371  PROGRAM SPECIFIC SUBROUTINES
;  372    CHECK_RD
;  393    CON_SET_FOR_CO
;  414    WCS_SIZER
;  488    WCS_SIZE_P
;  509    LOAD_MOD_DAT
;  524    TEST2E_ERROR
;  599  TEST 2E - WCS RAM MARCH PATTERN TEST
```

```
77
7a
7a
7a
```

```
7000    63
7000    64
7000    65  .TITLE  "ENKBF  8085 based diagnostic for WCS/DAP module  Rev 02.00"
7000    66  ;
7000    67  ;
7000    68  ;                                COPYRIGHT (c) 1982 BY
7000    69  ;                        DIGITAL EQUIPMENT CORPORATION, MAYNARD,
7000    70  ;                         MASSACHUSETTS.  ALL RIGHTS RESERVED.
7000    71  ;
7000    72  ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
7000    73  ; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
7000    74  ; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
7000    75  ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
7000    76  ; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
7000    77  ;
7000    78  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
7000    79  ; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
7000    80  ;
7000    81  ; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
7000    82  ; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
7000    83  ;
7000    84
7000    85  ;++
7000    86  ; FACILITY:      VAX-11/730 MICRO-DIAGNOSTIC.
7000    87  ;
7000    88  ; ABSTRACT:      This 8085 based diagnostic tests the hardcore logic of the
7000    89  ;                WCS and DAP modules.
7000    90  ;
7000    91  ; ENVIRONMENT:   ENKAA  Micro-diagnostic Monitor
7000    92  ;
7000    93  ; AUTHOR:        Jim Klumpp       4-MAY-81
7000    94  ;
7000    95  ; MODIFIED BY:   David Mayo       8-MAR-82        VERSION 01.00
7000    96  ;                Ernie Preisig    7-OCT-82        Version 02.00   RD Changes
7000   102  ;--
7000   103
7000   104
7000   105
7000   106
7000   107
7000   108
7000   109
7000   110
7000   111
7000   112
7000   113
7000   114
7000   115
7000   116
7000   117
7000   118
7000   119
7000   120
7000   121
7000   122
7000   123
7000   124
7000   125
7000   126
7000   127
```

```
7000  128
7000  129
7000  130
7000  131
7000  132 000                                  .PSECT  CPU6,BYTE
7000  133                                      .ALIGN  BYTE
7000  134
7000  135                                  ;*****************************************************************************
7000  136                                  ;
7000  137                                  ; DATA AREA
7000  138                                  ;
7000  139                                  ;*****************************************************************************
7000  140
7000  141 00000000                        HEAD            =0
7000  142
7000  143 00000000                                        EQUATE                  ;REGISTER EQUATE MACRO
            00000002
            00000004
            00000006
            00000006
            00000001
            00000003
            00000005
            00000007 00000006
7000  144
7000  145 00000000                        UPC14A          =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000  146
7000  147
7000  148 00000001                        MISC2           =<^X01>
7000  149 00000001                        BOOTF           =<^X01>                 ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000  150
7000  151
7000  152 00000002                        DCLO            =<^X02>                 ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000  153
7000  154 00000002                        HLTFLG          =<^X02>                 ;(RO) HALT FLAG ASSERTED LOW   <LSB>
7000  155
7000  156
7000  157 00000003                        READJ1          =<^X03>                 ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000  158
7000  159 00000003                        ALLOWP          =<^X03>                 ;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000  160
7000  161
7000  162 00000004                        READJ2          =<^X04>                 ;(RO) BIT 07 EQUALS J2 <MSB>
7000  163 00000004                        APTFLG          =<^X04>                 ;(RO) APT PRESENT FLAG <LSB>
7000  164
7000  165 00000005                        AUTO            =<^X05>                 ;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000  166
7000  167 00000006                        REMDIS          =<^X06>                 ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000  168
7000  169 00000007                        OKV15           =<^X07>                 ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000  170
7000  171 00000007                        BOOTEN          =<^X07>                 ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
7000  172
7000  173
7000  174 00000008                        WCSB0           =<^X08>                 ;(WO) WCS WRITE DATA REGISTER
7000  175 00000009                        WCSB1           =<^X09>
7000  176 0000000A                        WCSB2           =<^X0A>
7000  177
7000  178 0000001C                        ITDB            =<^X1C>                 ;(R/W) TIMER DATA REGISTER
7000  179 0000001D                        ITCSR           =<^X1D>                 ;(R/W) TIMER STATUS REGISTER
```

```
7000  180
7000  181 00000020                CLRCLK      =<^X20>     ;(WO) CLEAR CPU CLOCK RUN
7000  182 00000021                SETCLK      =<^X21>     ;(WO) SET CPU CLOCK RUN
7000  183
7000  184 00000022                CLRSS       =<^X22>     ;(WO) CLEAR CLOCK SINGLE STEP
7000  185 00000023                SETSS       =<^X23>     ;(WO) SET CLOCK SINGLE STEP
7000  186
7000  187 00000024                CLRCSK      =<^X24>     ;(WO) CLEAR CSR CLOCK
7000  188 00000025                SETCSK      =<^X25>     ;(WO) SET CSR CLOCK
7000  189
7000  190 00000026                CLRUPK      =<^X26>     ;(WO) CLEAR THE UPC CLOCK
7000  191 00000027                SETUPK      =<^X27>     ;(WO) SET THE UPC CLOCK
7000  192
7000  193 00000028                SETMCI      =<^X28>     ;(WO) SET MEMORY CONTROL INIT
7000  194 00000029                CLRMCI      =<^X29>     ;(WO) CLEAR MEMORY CONTROL INIT
7000  195
7000  196 0000002A                SETMCK      =<^X2A>     ;(WO) SET MEMORY CONTROL CLOCK
7000  197 0000002B                CLRMCK      =<^X2B>     ;(WO) CLEAR MEMORY CONTROL CLOCK
7000  198
7000  199 0000002C                CLRTMR      =<^X2C>     ;(WO) STOP TIMER
7000  200 0000002D                SETTMR      =<^X2D>     ;(WO) START TIMER COUNTING
7000  201
7000  202 0000002E                CLRBSY      =<^X2E>     ;(WO) CLEAR UNIBUS BUS BUSY
7000  203 0000002F                SETBSY      =<^X2F>     ;(WO) SET UNIBUS BUS BUSY
7000  204
7000  205 00000030                SETDCL      =<^X30>     ;(WO) SET DC LOW
7000  206 00000031                CLRDCL      =<^X31>     ;(WO) CLEAR DC LOW
7000  207
7000  208 00000032                SETACL      =<^X32>     ;(WO) SET AC LOW
7000  209 00000033                CLRACL      =<^X33>     ;(WO) CLEAR AC LOW
7000  210
7000  211 00000034                INITL       =<^X34>     ;(WO) CLEAR UNIBUS INIT
7000  212 00000035                INITH       =<^X35>     ;(WO) SET UNIBUS INIT
7000  213
7000  214 00000036                CLRWCK      =<^X36>     ;(WO) CLEAR WCS BIT 00
7000  215 00000037                SETWCK      =<^X37>     ;(WO) SET WCS BIT 00
7000  216
7000  217 00000038                CLRCSR      =<^X38>     ;(WO) CLEAR WCS BIT 01
7000  218 00000039                SETCSR      =<^X39>     ;(WO) SET WCS BIT 01
7000  219
7000  220 0000003C                CLRLIT      =<^X3C>     ;(WO) CLEAR THE RUN LIGHT
7000  221 0000003D                SETLIT      =<^X3D>     ;(WO) SET THE RUN LIGHT
7000  222
7000  223 00000044                TUDB        =<^X44>     ;(R/W) TU-58 DATA BUFFER
7000  224 00000045                TUSR        =<^X45>     ;(RO)  TU-58 STATUS REGISTER
7000  225 00000046                TUMODE      =<^X46>     ;(R/W) TU-58 MODE REGISTER 1 + 2
7000  226                                                  ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  227                                                  ;READING THE CR FORCES THE INTERNAL
7000  228                                                  ;POINTER BACK TO MR1
7000  229 00000047                TUCR        =<^X47>     ;(R/W) TU-58 COMMAND REGISTER
7000  230
7000  231 00000048                TTDB        =<^X48>     ;(R/W) TERMINAL DATA BUFFER
7000  232 00000049                TTSR        =<^X49>     ;(RO)  TERMINAL STATUS REGISTER
7000  233 0000004A                TTMODE      =<^X4A>     ;(R/W) TERMINAL MODE REGISTER 1 + 2
7000  234                                                  ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  235                                                  ;READING THE CR FORCES THE INTERNAL
7000  236                                                  ;POINTER BACK TO MR1
7000  237 0000004B                TTCR        =<^X4B>     ;(R/W) TERMINAL COMMAND REGISTER
7000  238
7000  239 00000080                READ        =<^X80>     ;(RO) CONSOLE READ REGISTER
```

```
ZZ-ENKBF-2.0    7000    4                                                Fiche 1  Frame L1          Sequence 11
7000   240
7000   241 00000082                    CPUACK          =<^X82>              ;(RO) ACKNOWLEDGE FROM CPU        <LSB>
7000   242
7000   243 00000083                    CPATTN          =<^X83>              ;(RO) ATTENTION FROM CPU          <LSB>
7000   244
7000   245 00000084                    UPC14           =<^X84>              ;(RO) UPC BIT 14                  <LSB>
7000   246
7000   247 00000085                    CSR07           =<^X85>              ;(RO) CSR BIT 7                   <LSB>
7000   248
7000   249 00000086                    CSR15           =<^X86>              ;(RO) CSR BIT 15                  <LSB>
7000   250
7000   251 00000087                    CSR23           =<^X87>              ;(RO) CSR BIT 23                  <LSB>
7000   252
7000   253 000000A0                    ENBPAR          =<^XA0>              ;(WO) ENABLE STALL ON PARITY ERROR
7000   254 000000A1                    DISPAR          =<^XA1>              ;(WO) DISABLE STALL ON PARITY ERROR
7000   255
7000   256 000000A2                    VSHIFT          =<^XA2>              ;(WO) SET THE V-BUS TO SHIFT MODE
7000   257 000000A3                    VNORML          =<^XA3>              ;(WO) SET THE V-BUS TO NORMAL MODE
7000   258
7000   259 000000A4                    SETTIM          =<^XA4>              ;(WO) SET THE INTERVAL TIMER INTERRUPT
7000   260 000000A5                    CLRTIM          =<^XA5>              ;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000   261
7000   262 000000A6                    ENBMEM          =<^XA6>              ;(WO) ENABLE MEMORY REFERENCES
7000   263 000000A7                    DISMEM          =<^XA7>              ;(WO) DISABLE MEMORY REFERENCES
7000   264
7000   265 000000A8                    SETACK          =<^XA8>              ;(WO) SET CONSOLE ACKNOWLEDGE
7000   266 000000A9                    CLRACK          =<^XA9>              ;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000   267
7000   268                             ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
7000   269                             ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
7000   270                             ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
7000   271
7000   272 000000A8                    CLRUPC          =<^XA8>              ;(WO) CLEAR THE V-BUS SERIAL INPUT
7000   273 000000A9                    SETUPC          =<^XA9>              ;(WO) SET THE V-BUS SERIAL INPUT
7000   274
7000   275 000000AA                    SETATN          =<^XAA>              ;(WO) SET CONSOLE ATTENTION
7000   276 000000AB                    CLRATN          =<^XAB>              ;(WO) CLEAR CONSOLE ATTENTION
7000   277
7000   278 000000AC                    SETPFI          =<^XAC>              ;(WO) SET POWER FAIL INTERRUPT
7000   279 000000AD                    CLRPFI          =<^XAD>              ;(WO) CLEAR POWER FAIL INTERRUPT
7000   280
7000   281 000000AE                    SETHLT          =<^XAE>              ;(WO) SET THE CONSOLE HALT BIT
7000   282 000000AF                    CLRHLT          =<^XAF>              ;(WO) CLEAR THE CONSOLE HALT BIT
7000   283
7000   284 000000CC                    WRITE           =<^XCC>              ;(WO) THE CONSOLE WRITE REGISTER
7000   285
7000   286 000000EC                    YBUSRD          =<^XEC>              ;(RO) Y-BUS READ
7000   287
7000   288
7000   289                             ;ADDITIONAL DATA
7000   290
7000   291 00004C22                    PARITY_VECTOR   = <^X4C22>           ;PARITY ERROR BRANCH VECTOR.
7000   292 00004C25                    POWER_VECTOR    = <^X4C25>           ;POWER FAIL BRANCH VECTOR.
7000   293 000040D9                    MIPFLG          =<^X40D9>            ;ADDRESS OF MODEM IN PROGRESS FLAG
7000   294
7000   295
7000   296                             ;*************************************************************************************
7000   297                             ;
7000   298                             ;  PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
7000   299                             ;
```

```
7000  300
7000  301
7000  302 015B'C3                  ENTRY:          JMP     TEST2E
7003  303
7003  304                          ;*****************************************************************************
7003  305                          ;
7003  306                          ;  VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
7003  307                          ;  BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
7003  308                          ;
7003  309                          ;*****************************************************************************
7003  310
7003  311 0200                    VERSION:        .WORD           <^X0200>        ;VERSION NUMBER (REV 02.00)
7005  312 46 42                   FILE_NAME:      .ASCII          'BF'            ;LAST 2 CHARS OF FILE NAME
7007  313
7007  314                          ;*****************************************************************************
7007  315                          ;
7007  316                          ; VARIABLES AREA
7007  317                          ;
7007  318                          ;*****************************************************************************
7007  319
7007  320
7007  321 FF3F                    ADDR_16K:       .WORD           <^XFF3F>        ;ADDRESS OF 16K.
7009  322 0000000B                WCS_SIZE:       .BLKB           2               ;HIGHEST WCS ADDRESS.
700B  323 0000000D                WCS_ADDR_PNT:   .BLKB           2               ;POINTS TO WCS_SIZER_ADDR.
700D  324 00                      LOOP_CNT:       .BYTE           0               ;COUNTS LOOP PASSES FOR ^C USE.
700E  325 0000                    SAVE_WCS_ADD:   .WORD           0               ;SAVE WCS ADDRESS ON ^C.
7010  326
7010  327 003C                    WCS_SIZER_ADDR: .WORD           <^X003C>        ;*
7012  328 0040                                    .WORD           <^X0040>        ;WCS ADDRESSES TO BE WRITTEN TO
7014  329 0044                                    .WORD           <^X0044>        ; DETERMINE IF WCS IS 16K,17K,
7016  330 0048                                    .WORD           <^X0048>        ; 18K,19K OR 20K.
7018  331 004C                                    .WORD           <^X004C>        ;*
701A  332
701A  333 11                      ADD_WCS_DAT:    .BYTE           <^X11>          ;DATA TO TEST ADDITIONAL
701B  334 11                                      .BYTE           <^X11>          ; WCS GREATER THAN 16K, WHICH
701C  335 11                                      .BYTE           <^X11>          ; CONSISTS OF FOUR BIT BLOCKS
701D  336 55                                      .BYTE           <^X55>          ; OF A ONE THROUGH A ZERO FIELD
701E  337 55                                      .BYTE           <^X55>          ; AND A ZERO THROUGH A ONE
701F  338 55                                      .BYTE           <^X55>          ; FIELD.
7020  339
7020  340 20                      HEADER_B:       .BYTE           32              ;32 CHARACTERS FOLLOW...
7021  341 48 47 49 48 00'                         .ASCII          <CR>'HIGHEST WCS ADRRESS AVAILABLE: '
            57 20 54 53 45
            44 41 20 53 43
            53 53 45 52 52
            49 41 56 41 20
            45 4C 42 41 4C
            20 3A
7041  342
7041  343
7041  344 00                      ZERO_DAT:       .BYTE           0               ;*
7042  345 00                                      .BYTE           0               ;*
7043  346 00                                      .BYTE           0               ;STANDARD DATA PATTERNS.
7044  347 FF                    ONE_DAT:        .BYTE           <^XFF>          ;*
7045  348 FF                                    .BYTE           <^XFF>          ;*
7046  349 FF                                    .BYTE           <^XFF>          ;*
7047  350
7047  351
7047  352                          ;*****************************************************************************
7047  353                          ;
```

```
7047  354
7047  355
7047  356
7047  357
7047  358
7047  359 03                  EXP_SHIFT:      .BYTE   3               ;LOCATION OF EXPECTED DATA FROM
7048  360 0000004B            EXP_DAT:        .BLKB   3               ; VARIOUS TESTS.
704B  361
704B  362 03                  REC_SHIFT:      .BYTE   3               ;LOCATION OF RETURNED DATA FROM
704C  363 0C000004F           REC_DAT:        .BLKB   3               ; VARIOUS TESTS.
704F  364
704F  365
704F  366
704F  367
704F  368                     ;  CHECK_RD     THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
704F  369                     ;                IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
704F  370                     ;                KEEP TRACK OF ELAPSED TIME.
704F  371
704F  372
704F  373 40D9 3A             CHECK_RD:       LDA     MIPFLG          ;GET STATE OF MODEM IN PROGRESS
7052  374 B7                                  ORA     A               ;SET CONDITION CODES
7053  375 C8                                  RZ                      ;RETURN IF ZERO
7054  376 0000'CD                             CALL    GCVECT          ;ELSE GO TO BOOT CODE IF RD
7057  377
7057  378 0000'3A                             LDA     CNTLC_TYPED     ;SEE IF CONTROL C TYPED (UNDER RD)
705A  379 0F                                  RRC
705B  380 005F'DC                             CC      CON_SET_FOR_CO   ;IF SO, GO TO MONITOR.
705E  381
705E  382 C9                                  RET                     ;BACK TO TEST
705F  383
705F  384
705F  385
705F  386
705F  387                     ;  CON_SET_FOR_CO  THIS ROUTINE IS USED IF A CONTROL C OCCURS TO SAVE THE
705F  388                     ;                  VALUE OF WCS_ADDRESS. IT THEN RESTOES THIS VALUE IF A
705F  389                     ;                  CONTINUE IS ISSUED.
705F  390
705F  391
705F  392
705F  393
705F  394 0000'2A             CON_SET_FOR_CO: LHLD    WCS_ADDRESS     ;SAVE THE PRESENT WCS ADDRESS.
7062  395 000E'22                             SHLD    SAVE_WCS_ADD
7065  396
7065  397 0000'CD                             CALL    SET_FOR_CONT    ;RETURN TO MONITOR.
7068  398
7068  399 000E'2A                             LHLD    SAVE_WCS_ADD    ;RESTORE STATE.
706B  400 0000'22                             SHLD    WCS_ADDRESS
706E  401
706E  402 C9                                  RET
706F  403
706F  404
706F  405
706F  406
706F  407                     ;  WCS_SIZER    THIS ROUTINE SIZES THE WCS BY WRITING DATA PATTERNS AT 16K,
706F  408                     ;                17K, 18K, 19K AND 20K AND THEN READING THESE PATTERNS BACK. IF
706F  409                     ;                THE PATTERN RECEIVED IS THE SAME AS THE ONE SENT OUT THEN WCS
706F  410                     ;                IS ASSUMED TO EXIST AT THAT LOCATION.
706F  411
706F  412
706F  413
```

```
706F  414
706F  415 0010'2A                          WCS_SIZER:    LHLD    WCS_SIZER_ADDR  ;SET INITIAL WCS SIZE TO 16K.
7072  416 0009'22                                        SHLD    WCS_SIZE
7075  417
7075  418 00'10'21                                       LXI     H,WCS_SIZER_ADDR        ;SIZE WCS AT 16K.
7078  419 000B'22                                        SHLD    WCS_ADDR_PNT
707B  420
707B  421 C5 46 00'00'21       ******************      DO      5
          77 05 3E             *
7083  422                      *
7083  423 000B'2A             *                         LHLD    WCS_ADDR_PNT    ;PREPARE TO WRITE TO WCS.
7086  424 00'00'11             *                         LXI     D,WCS_ADDRESS
7089  425 02 0E                *                         MVI     C,2
708B  426 0000'CD              *                         CALL    MOVER
708E  427                      *
708E  428 00'41'21             *                         LXI     H,ZERO_DAT      ;LOAD DATA TO BE WRITTEN TO WCS
7091  429 00'00'11             *                         LXI     D,WCS_VALUE     ; INTO WCS_VALUE.
7094  430 0000'CD              *                         CALL    MOVER_3
7097  431                      *
7097  432 0000'CD              *                         CALL    WCS_WRITE       ;WRITE TO WCS.
709A  433                      *
709A  434 0000'CD              *                         CALL    WCS_READ        ;READ FROM WCS.
709D  435                      *
709D  436 00'00'21             * ****************        IFEQ    WCS_VALUE,ONE_DAT,3    ;IF RECEIVED DATA IS ALL ONES...
          03 0E 00'44'11       * *
          00AE'C2 0000'CD      * *
70AB  437                      * *
70AB  438 00D7'C3              * *                       JMP     1$              ;...LEAVE WCS_SIZE AS IS.
70AE  439                      * *
70AE  440                      * ****************        ENDIF
70AE  441                      *
70AE  442 004F'CD              *                         CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
70B1  443                      *
70B1  444 00'41'21             *                         LXI     H,ZERO_DAT          ;LOAD DATA TO BE WRITTEN TO WCS
70B4  445 00'00'11             *                         LXI     D,WCS_VALUE         ; INTO WCS_VALUE.
70B7  446 0000'CD              *                         CALL    MOVER_3
70BA  447                      *
70BA  448 0000'CD              *                         CALL    WCS_WRITE       ;WRITE TO WCS.
70BD  449                      *
70BD  450 0000'CD              *                         CALL    WCS_READ        ;READ FROM WCS.
70C0  451                      *
70C0  452 00'00'21             * ****************        IFEQ    WCS_VALUE,ONE_DAT,3    ;IF RECEIVED DATA IS ALL ONES...
          03 0E 00'44'11       * *
          00D1'C2 0000'CD      * *
70CE  453                      * *
70CE  454 00D7'C3              * *                       JMP     1$              ;...LEAVE WCS_SIZE AS IS.
70D1  455                      * *
70D1  456                      * ****************        ENDIF
70D1  457                      *
70D1  458 0000'2A              *                         LHLD    WCS_ADDRESS     ;PRESENT WCS LOCATION EXISTS. SET
70D4  459 0009'22              *                         SHLD    WCS_SIZE        ; WCS SIZE TO THIS LOCATION.
70D7  460                      *
70D7  461 000B'2A              *                 1$:     LHLD    WCS_ADDR_PNT    ;TRY THE NEXT HIGHEST WCS LOCATION.
70DA  462 23                   *                         INX     H
70DB  463 23                   *                         INX     H
70DC  464 000B'22              *                         SHLD    WCS_ADDR_PNT
70DF  465                      *
70DF  466 004F'CD              *                         CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
70E2  467                      *
70E2  468 35 00'00'21          ******************        ENDDO
```

```
                70 C1 0083'C2
70EB  469
70EB  470 0009'3A                       LDA     WCS_SIZE        ;SET BITS 8 AND 9 OF WCS_SIZE TO
70EE  471 03 F6                         ORI     3               ; DISPLAY THE HIGHEST WCS ADDRESS
70F0  472 0009'32                       STA     WCS_SIZE        ; POSSIBLE.
70F3  473
70F3  474 000A'3A                       LDA     WCS_SIZE+1      ;SET BITS 0-7 OF WCS_SIZE.
70F6  475 FF F6                         ORI     <^XFF>
70F8  476 000A'32                       STA     WCS_SIZE+1
70FB  477
70FB  478 C9                            RET
70FC  479
70FC  480
70FC  481                       ;*****************************************************************************
70FC  482                       ;
70FC  483                       ;  WCS_SIZE_P  THIS ROUTINE IS USED AFTER THE WCS_SIZER ROUTINE TO
70FC  484                       ;              PRINT THE HIGHEST WCS ADDRESS AVAILABLE.
70FC  485                       ;
70FC  486                       ;*****************************************************************************
70FC  487
70FC  488
70FC  489 00'20'21              WCS_SIZE_P:     LXI     H,HEADER_B      ;PRINT THE WCS SIZE MESSAGE.
70FF  490 0000'CD                               CALL    PRINT_STRING
7102  491
7102  492 00'09'21                             LXI     H,WCS_SIZE      ;PRINT THE HIGHEST WCS ADDRESS.
7105  493 00'00'11                             LXI     D,HEX_BUF
7108  494 02 0E                               MVI     C,2
710A  495 0000'CD                             CALL    MOVER
710D  496 02 0E                               MVI     C,2
710F  497 0000'CD                             CALL    PRINT_HEX
7112  498
7112  499 C9                                  RET
7113  500
7113  501
7113  502                       ;*****************************************************************************
7113  503                       ;
7113  504                       ;  LOAD_MOD_DAT  THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
7113  505                       ;                PAIR INTO CON_MOD_DAT.
7113  506                       ;
7113  507                       ;*****************************************************************************
7113  508
7113  509
7113  510 00'00'11              LOAD_MOD_DAT:   LXI     D,CON_MOD_DAT   ;LOAD THE MODULE DATA POINTED TO BY
7116  511 05 0E                               MVI     C,5             ; THE H,L PAIR INTO CON_MOD_DAT.
7118  512 0000'CD                             CALL    MOVER
711B  513
711B  514 C9                                  RET
711C  515
711C  516
711C  517                       ;*****************************************************************************
711C  518                       ;
711C  519                       ;  TEST2E_ERROR  THIS ROUTINE IS USED BY TEST 2E TO DEFINE THE ERROR VARIABLES
711C  520                       ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
711C  521                       ;
711C  522                       ;*****************************************************************************
711C  523
711C  524
711C  525 00'48'21              TEST2E_ERROR:   LXI     H,EXP_DAT       ;MOVE EXPECTED DATA TO CON_EXP_DAT.
711F  526 00'01'11                            LXI     D,CON_EXP_DAT+1
7122  527 0000'CD                             CALL    MOVER_3
```

```
ZZ-ENKBF-2.0    7000    4                                   Fiche 1  Frame D2          Sequence 16
7125   528
7125   529 00'00'21                        LXI     H,WCS_VALUE     ;MOVE RECEIVED DATA TO CON_REC_DAT.
7128   530 00'01'11                        LXI     D,CON_REC_DAT+1
712B   531 0000'CD                         CALL    MOVER_3
712E   532
712E   533 FF 3E                           MVI     A,<^XFF>        ;LAST THREE BYTES ARE OF INTEREST.
7130   534 0000'32                         STA     CON_MSK_DAT
7133   535 03 0E 00'01'21                  ZERO    CON_MSK_DAT+1,3
           0D 23 77 AF
           0139'C2
713F   536
713F   537 02 0E 00'00'21                  ZERO    CON_ADD_DAT,2   ;ADDITIONAL DATA CONTAINS THE
           0D 23 77 AF
           0145'C2
714B   538 0000'2A                         LHLD    WCS_ADDRESS     ; PRESENT WCS ADDRESS.
714E   539 0002'22                         SHLD    CON_ADD_DAT+2
7151   540
7151   541 00'00'21                        LXI     H,MWCS_A        ;WCS MODULE BEING TESTED.
7154   542 0113'CD                         CALL    LOAD_MOD_DAT
7157   543
7157   544 0000'CD                         CALL    CON_ERROR       ;CALL ERROR ROUTINE.
715A   545
715A   546 C9                              RET
715B   547
715B   548
715B   549
715B   550
715B   551
715B   552
715B   553
715B   554
715B   555
715B   556
715B   557
715B   558
715B   559
715B   560
715B   561
715B   562
715B   563
715B   564
715B   565
715B   566
715B   567
715B   568
715B   569
715B   570
715B   571
715B   572
715B   573
715B   574
715B   575
715B   576
715B   577
715B   578
715B   579
715B   580
715B   581
715B   582
715B   583
```

```
;*******************************************************************************
;
;       TEST2E          WCS RAM MARCH PATTERN TEST
;       ------          --------------------------
;
;               THE WCS RAM MARCH PATTERN TEST WRITES A BACKGROUND OF ZEROS
;       THROUGH THE MEMORY THEN GOES TO EACH LOCATION WORKING ITS WAY UP
;       READING THE 0 BACKGROUND WRITING A 1 AND READING THE ONE BACK.
;       THIS THEN LEAVES 1'S IN THE BACKGROUND AND THE TEST WILL START
;       AT THE HIGH ADDRESS AND READ 1'S, WRITE 0'S, READ 0'S.
;                THE MEMORY WILL BE SIZED BY WRITING INTO HIGH ADDRESSES
;       AND READING AT THESE ADDRESSES TO SEE IF THE DATA HAS BEEN WRITTEN
;       AT THESE LOCATIONS.
;
;
;       TEST STEPS:
;
;               1)  SET ADDRESS IN UPC.
;               2)  WRITE BACKGROUND OF ZEROS.
;               3)  START AT LOCATION ZERO, READ AND COMPARE TO ZERO.
;               4)  PRINT ERROR IF ANY.
;               5)  WRITE 1, READ  AND COMPARE TO 1.
;               6)  PRINT ERROR IF ANY.
;               7)  REPEAT 3-6 FOR ALL LOCATIONS (LOW TO HIGH).
;               8)  STARTING AT TOP OF RAM, READ AND COMPARE TO ONE.
;               9)  PRINT ERROR IF ANY.
;               10) WRITE 0 READ  AND COMPARE TO 0.
;               11) PRINT ERROR IF ANY.
;               12) REPEAT 8-11 FOR ALL LOCATIONS (HIGH TO LOW).
;
;
;       ERROR1: WCS RAM MARCH TEST FAILURE ON FIRST PASS (UP).
;       ERROR2: WCS RAM MARCH TEST FAILURE ON SECOND PASS (DOWN).
```

```
715B  584
715B  585
715B  586
715B  587
715B  588
715B  589
715B  590
715B  591
715B  592
715B  593
715B  594
715B  595
715B  596
715B  597
715B  598
715B  599
715B  600 0000'CD 2E 3E                                  TEST2E:         HEAD    <^X2E>
          0F 0000'3A
          C9 0168'D2
          0F 0000'3A
          3D D3 02F1'DA
7171  601
7171  602 006F'CD                                                        CALL    WCS_SIZER       ;SIZE WCS.
7174  603
7174  604 00FC'CD                                                        CALL    WCS_SIZE_P      ;PRINT THE WCS SIZE.
7177  605
7177  606 FF 3E                                                         MVI     A,<^XFF>        ;INITIALIZE LOOP COUNT (FOR ^C AND RD).
7179  607 000D'32                                                       STA     LOOP_CNT
717C  608
717C  609 0009'2A                                                       LHLD    WCS_SIZE        ;POINT TO THE HIGHEST WCS LOCATION.
717F  610 0000'22                                                       SHLD    WCS_ADDRESS
7182  611
7182  612 03 0E 00'00'21                                               ZERO    WCS_VALUE,3     ;CLEAR WCS_VALUE.
          0D 23 77 AF
          0188'C2
718E  613
718E  614 0000'CD                                        1$:            CALL    WCS_WRITE       ;WRITE A ZERO TO WCS.
7191  615
7191  616 004F'CD                                                        CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7194  617
7194  618 00'0D'21                                                     LXI     H,LOOP_CNT      ;POINT TO LOOP_CNT.
7197  619 35                                                           DCR     M               ;DECREMENT LOOP_CNT.
7198  620 01AA'C2                                                      JNZ     2$              ;IF NOT ZERO, DON'T CHECK FOR ^C.
719B  621
719B  622 FF 3E                                                        MVI     A,<^XFF>        ;INITIALIZE LOOP_CNT.
719D  623 000D'32                                                      STA     LOOP_CNT
71A0  624
71A0  625 0000'CD                                                      CALL    GCVECT          ;SEE IF A CONTROL C HAS BEEN
71A3  626 0000'3A                                                      LDA     CNTLC_TYPED     ; TYPED.
71A6  627 0F                                                           RRC
71A7  628 005F'DC                                                      CC      CON_SET_FOR_CO  ;IF SO, GO TO THE MONITOR.
71AA  629
71AA  630 00'00'21                       ********** 2$:               IFEQ    WCS_ADDRESS,ZERO_DAT,2   ;COMPARE THE WCS POINTER TO 0.
          02 0E 00'41'11                 *
          01BB'C2 0000'CD                *
71B8  631                                *
71B8  632 01C4'C3                        *                             JMP     3$              ;FIRST WCS PASS IS COMPLETED.
71BB  633                                *
71BB  634                                **********                    ENDIF
71BB  635
```

;NOTE: ADDITIONAL DATA CONTAINS THE PRESENT WCS ADDRESS.
;
;DEBUG: 1) CHECK "CS ADRS H","ROW ADRS STB L","COL ADRS STB L"
;   FOR THE ADDRESS BEING USED.
;   2) CHECK REFRESH LOGIC FOR "REFR REQ L","REFR CYCLE H"
;   AND "REFR ADDRS H".
;   3) CHECK FOR EACH "NAD L" BUS SIGNAL TO REACH THE
;   REFRESH LOGIC.
;   4) CHECK RAM MEMORY FOR ERRORS IN INTERNAL ADDRESSS LOGIC.
;
;***

```
71BB  636 00'00'21                                  LXI     H,WCS_ADDRESS   ;DECREMENT THE WCS POINTER.
71BE  637 0000'CD                                   CALL    DECR_WORD
71C1  638
71C1  639 018E'C3                                   JMP     1$              ;LOAD THE NEXT WCS LOCATION WITH ZERO.
71C4  640
71C4  641
71C4  642 01 3E                           3$:       MVI     A,1             ;FIRST PART OF TEST 6.
71C6  643 0000'32                                   STA     CON_ERR_NUM
71C9  644
71C9  645 FF 3E                                     MVI     A,<^XFF>        ;INITIALIZE LOOP_CNT.
71CB  646 000D'32                                   STA     LOOP_CNT
71CE  647
71CE  648 004F'CD                         4$:       CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
71D1  649
71D1  650 0000'CD                                   CALL    WCS_READ        ;READ WCS.
71D4  651
71D4  652 00'41'21              ******************  IFNEQ   ZERO_DAT,WCS_VALUE,3    ;COMPARE THE EXPECTED AND
          03 0E 00'00'11        *
          01F1'CA 0000'CD       *
71E2  653                       *                                               ; RECEIVED DATA.
71E2  654                       *
71E2  655 03 0E 00'48'21        *                 ZERO    EXP_DAT,3       ;LOAD ALL ZEROS INTO EXP_DAT.
          0D 23 77 AF           *
          01E8'C2               *
71EE  656                       *
71EE  657 011C'CD               *                 CALL    TEST2E_ERROR    ;PRINT AN ERROR MESSAGE.
71F1  658                       *
71F1  659                       ******************  ENDIF
71F1  660
71F1  661 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
71F4  662 0F                                        RRC
71F5  663 01CE'DA                                   JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
71F8  664
71F8  665 00'44'21                        5$:       LXI     H,ONE_DAT       ;LOAD ALL ONES INTO WCS_VALUE.
71FB  666 00'00'11                                  LXI     D,WCS_VALUE
71FE  667 0000'CD                                   CALL    MOVER_3
7201  668
7201  669 0000'CD                                   CALL    WCS_WRITE       ;WRITE TO WCS.
7204  670
7204  671 0000'CD                                   CALL    WCS_READ        ;READ FROM WCS.
7207  672
7207  673 00'44'21              ******************  IFNEQ   ONE_DAT,WCS_VALUE,3         ;COMPARE THE EXPECTED AND
          03 0E 00'00'11        *
          0221'CA 0000'CD       *
7215  674                       *                                                     ; RECEIVED DATA.
7215  675                       *
7215  676 00'44'21             *                 LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7218  677 00'48'11              *                 LXI     D,EXP_DAT
721B  678 0000'CD               *                 CALL    MOVER_3
721E  679                       *
721E  680 011C'CD               *                 CALL    TEST2E_ERROR    ;PRINT AN ERROR MESSAGE.
7221  681                       *
7221  682                       ******************  ENDIF
7221  683
7221  684 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7224  685 0F                                        RRC
7225  686 01F8'DA                                   JC      5$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7228  687
7228  688 00'0D'21                                  LXI     H,LOOP_CNT      ;POINT TO LOOP_CNT.
722B  689 35                                        DCR     M               ;DECREMENT LOOP_CNT.
```

```
722C  690 023E'C2                                    JNZ     6$              ;IF NOT ZERO, DON'T CHECK FOR ^C.
722F  691
722F  692 FF 3E                                      MVI     A,<^XFF>        ;INITIALIZE LOOP_CNT.
7231  693 000D'32                                    STA     LOOP_CNT
7234  694
7234  695 0000'CD                                    CALL    GCVECT          ;SEE IF A CONTROL C HAS BEEN
7237  696 0000'3A                                    LDA     CNTLC_TYPED     ; TYPED.
723A  697 0F                                         RRC
723B  698 005F'DC                                    CC      CON_SET_FOR_CO  ;IF SO, GO TO THE MONITOR.
723E  699
723E  700 00'09'21          ****************** 6$:  IFEQ    WCS_SIZE,WCS_ADDRESS,2   ;CHECK TO SEE IF WCS POINTER
          02 0E 00'00'11    *
          024F'C2 0000'CD   *
724C  701                   *                                                     ; CONTAINS HIGHEST WCS LOCAT.
724C  702                   *
724C  703 0258'C3            *                      JMP     7$              ;SECOND WCS PASS IS COMPLETED.
724F  704                   *
724F  705                   ******************     ENDIF
724F  706
724F  707 00'00'21                                   LXI     H,WCS_ADDRESS   ;INCREMENT THE WCS POINTER.
7252  708 0000'CD                                    CALL    INC_WORD
7255  709
7255  710 01CE'C3                                    JMP     4$              ;TEST NEXT WCS LOCATION.
7258  711
7258  712 02 3E                              7$:     MVI     A,2             ;SECOND PART OF TEST 6.
725A  713 0000'32                                    STA     CON_ERR_NUM
725D  714
725D  715 FF 3E                                      MVI     A,<^XFF>        ;INITIALIZE LOOP_CNT.
725F  716 000D'32                                    STA     LOOP_CNT
7262  717
7262  718 004F'CD                              8$:     CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7265  719
7265  720 0000'CD                                    CALL    WCS_READ        ;READ WCS.
7268  721
7268  722 00'44'21                ****************** IFNEQ   ONE_DAT,WCS_VALUE,3     ;COMPARE THE EXPECTED AND
          03 0E 00'00'11          *
          0282'CA 0000'CD         *
7276  723                         *                                                 ; RECEIVED DATA.
7276  724                         *
7276  725 00'44'21               *                LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7279  726 00'48'11               *                LXI     D,EXP_DAT
727C  727 0000'CD                *                CALL    MOVER_3
727F  728                        *
727F  729 011C'CD                *                CALL    TEST2E_ERROR    ;PRINT AN ERROR MESSAGE.
7282  730                        *
7282  731                        ******************     ENDIF
7282  732
7282  733 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7285  734 0F                                         RRC
7286  735 0262'DA                                    JC      8$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7289  736
7289  737 03 0E 00'00'21                     9$:     ZERO    WCS_VALUE,3     ;CLEAR WCS VALUE.
          0D 23 77 AF
          028F'C2
7295  738
7295  739 0000'CD                                    CALL    WCS_WRITE       ;WRITE TO WCS.
7298  740
7298  741 0000'CD                                    CALL    WCS_READ        ;READ FROM WCS.
729B  742
729B  743 00'41'21          ******************     IFNEQ   ZERO_DAT,WCS_VALUE,3    ;COMPARE THE EXPECTED AND
```

```
                                                                                                    ; RECEIVED DATA.
72A9  744                                                                                                                
72A9  745                                                                                                                
72A9  746 03 0E 00'48'21                *                   ZERO    EXP_DAT,3       ;CLEAR EXP_DAT.
          0D 23 77 AF                   *
          02AF'C2                       *
72B5  747                               *
72B5  748 011C'CD                       *                   CALL    TEST2E_ERROR    ;PRINT AN ERROR MESSAGE.
72B8  749                               *
72B8  750                               ********************ENDIF
72B8  751
72B8  752 0000'3A                                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
72BB  753 0F                                                RRC
72BC  754 0289'DA                                           JC      9$              ;...JUMP TO BEGINNING OF ERROR LOOP.
72BF  755
72BF  756 00'0D'21                                          LXI     H,LOOP_CNT      ;POINT TO LOOP_CNT.
72C2  757 35                                                DCR     M               ;DECREMENT LOOP_CNT.
72C3  758 02D5'C2                                           JNZ     10$             ;IF NOT ZERO, DON'T CHECK FOR ^C.
72C6  759
72C6  760 FF 3E                                             MVI     A,<^XFF>        ;INITIALIZE LOOP_CNT.
72C8  761 000D'32                                           STA     LOOP_CNT
72CB  762
72CB  763 0000'CD                                           CALL    GCVECT          ;SEE IF A CONTROL C HAS BEEN
72CE  764 0000'3A                                           LDA     CNTLC_TYPED     ; TYPED.
72D1  765 0F                                                RRC
72D2  766 005F'DC                                           CC      CON_SET_FOR_CO  ;IF SO, GO TO THE MONITOR.
72D5  767
72D5  768 00'00'21                      ********** 10$:     IFEQ    WCS_ADDRESS,ZERO_DAT,2    ;COMPARE WCS_ADDRESS TO 0.
          02 0E 00'41'11                *
          02E6'C2 0000'CD               *
72E3  769                               *
72E3  770 02EF'C3                       *                   JMP     11$             ;THIRD WCS PASS IS COMPLETED.
72E6  771                               *
72E6  772                               ********************ENDIF
72E6  773
72E6  774 00'00'21                                          LXI     H,WCS_ADDRESS   ;DECREMENT THE WCS POINTER.
72E9  775 0000'CD                                           CALL    DECR_WORD
72EC  776
72EC  777 0262'C3                                           JMP     8$              ;TEST NEXT WCS LOCATION.
72EF  778
72EF  779 3C D3                                 11$:        TAIL
72F1  780
72F1  781
72F1  782
72F1  783                                       ;*******************************************************************************
72F1  784                                       ;
72F1  785                                       ;  LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
72F1  786                                       ;
72F1  787                                       ;*******************************************************************************
72F1  788
72F1  789
72F1  790 0000'CD 51 3E                         END_TEST:   HEAD    <^X51>
          0F 0000'3A
          C9 02FE'D2
          0F 0000'3A
          3D D3 0309'DA
7307  791
7307  792 3C D3                                             TAIL
7309  793
```

```
$PSW            = 00000006
$SP             = 00000006
A               = 00000007
ADDR_16K          00000007 R      02
ADD_WCS_DAT       0000001A R      02
ALLOWP          = 00000003
APTFLG          = 00000004
AUTO            = 00000005
B               = 00000000
BOOTEN          = 00000007
BOOTF           = 00000001
C               = 00000001
CHECK_RD          0000004F R      02
CLRACK          = 000000A9
CLRACL          = 00000033
CLRATN          = 000000AB
CLRBSY          = 0000002E
CLRCLK          = 00000020
CLRCSK          = 00000024
CLRCSR          = 00000038
CLRDCL          = 00000031
CLRHLT          = 000000AF
CLRLIT          = 0000003C
CLRMCI          = 00000029
CLRMCK          = 0000002B
CLRPFI          = 000000AD
CLRSS           = 00000022
CLRTIM          = 000000A5
CLRTMR          = 0000002C
CLRUPC          = 000000A8
CLRUPK          = 00000026
CLRWCK          = 00000036
CNT             = 000007E4
CNTA            = 00000FA0
CNTLC_TYPED       ********    X   02
COMPARE           ********    X   02
CON_ADD_DAT       ********    X   02
CON_BEGIN_TEST    ********    X   02
CON_ERROR         ********    X   02
CON_ERR_NUM       ********    X   02
CON_EXP_DAT       ********    X   02
CON_MOD_DAT       ********    X   02
CON_MSK_DAT       ********    X   02
CON_REC_DAT       ********    X   02
CON_SET_FOR_CO    0000005F R      02
CPATTN          = 00000083
CPUACK          = 00000082
CR                ********    X   02
CSR07           = 00000085
CSR15           = 00000086
CSR23           = 00000087
D               = 00000002
DCLO            = 00000002
DECR_WORD         ********    X   02
DISMEM          = 000000A7
DISPAR          = 000000A1
DOLOOP            ********    X   02
E               = 00000003
ENBMEM          = 000000A6
ENBPAR          = 000000A0
END_TEST          000002F1 R      02
ENTRY             00000000 R      02
EOS_FLG           ********    X   02
ERROR_CON         ********    X   02
EXP_DAT           00000048 R      02
EXP_SHIFT         00000047 R      02
FILE_NAME         00000005 R      02
GCVECT            ********    X   02
H               = 00000004
HEAD            = 00000000
HEADER_B          00000020 R      02
HEX_BUF           ********    X   02
HLTFLG          = 00000002
INC_WORD          ********    X   02
INITH           = 00000035
INITL           = 00000034
ITCSR           = 0000001D
ITDB            = 0000001C
L               = 00000005
LITERAL         = 00000001
LOAD_MOD_DAT      00000113 R      02
LOOP_CNT          0000000D R      02
LVL             = 00000000
M               = 00000006
MACSTK1         = 000007E4
MACSTK2         = 000007E2
MIPFLG          = 000040D9
MISC2           = 00000001
MOVER             ********    X   02
MOVER_3           ********    X   02
MWCS_A            ********    X   02
OKV15           = 00000007
ONE_DAT           00000044 R      02
PARITY_VECTOR   = 00004C22
POWER_VECTOR    = 00004C25
PRINT_HEX         ********    X   02
PRINT_STRING      ********    X   02
READ            = 00000080
READJ1          = 00000003
READJ2          = 00000004
REC_DAT           0000004C R      02
REC_SHIFT         0000004B R      02
REMDIS          = 00000006
SAVE_WCS_ADD      0000000E R      02
SETACK          = 000000A8
SETACL          = 00000032
SETATN          = 000000AA
SETBSY          = 0000002F
SETCLK          = 00000021
SETCSK          = 00000025
SETCSR          = 00000039
SETDCL          = 00000030
SETHLT          = 000000AE
SETLIT          = 0000003D
SETMCI          = 00000028
SETMCK          = 0000002A
SETPFI          = 000000AC
SETSS           = 00000023
SETTIM          = 000000A4
SETTMR          = 0000002D
SETUPC          = 000000A9
SETUPK          = 00000027
SETWCK          = 00000037
SET_FOR_CONT      ********    X   02
SKIP_TEST         ********    X   02
SYM             = 000007E4
TEST2E            0000015B R      02
TEST2E_ERROR      0000011C R      02
TTCR            = 0000004B
TTDB            = 00000048
TTMODE          = 0000004A
TTSR            = 00000049
TUCR            = 00000047
TUDB            = 00000044
TUMODE          = 00000046
TUSR            = 00000045
UPC14           = 00000084
UPC14A          = 00000000
VERSION           00000003 R      02
VNORML          = 000000A3
VSHIFT          = 000000A2
WCSB0           = 00000008
WCSB1           = 00000009
WCSB2           = 0000000A
WCS_ADDRESS       ********    X   02
WCS_ADDR_PNT      0000000B R      02
WCS_READ          ********    X   02
WCS_SIZE          00000009 R      02
WCS_SIZER         0000006F R      02
WCS_SIZER_ADDR    00000010 R      02
WCS_SIZE_P        000000FC R      02
WCS_VALUE         ********    X   02
WCS_WRITE         ********    X   02
WRITE           = 000000CC
Y               = 00000057
YBUSRD          = 000000EC
ZERO_DAT          00000041 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CPU6	00000309 (777.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.05	00:00:00.63
Command processing	33	00:00:00.26	00:00:02.00
Pass 1	519	00:00:10.53	00:00:44.33
Symbol table sort	0	00:00:00.22	00:00:00.73
Pass 2	231	00:00:04.59	00:00:18.05
Symbol table output	13	00:00:00.11	00:00:00.36
Psect synopsis output	3	00:00:00.04	00:00:00.12
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	826	00:00:15.81	00:01:06.23

The working set limit was 900 pages.
90303 bytes (177 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 157 non-local and 32 local symbols.
2817 source lines were read in Pass 1, producing 20 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBF.LIS/OBJECT=ENKBF.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBF.MAC